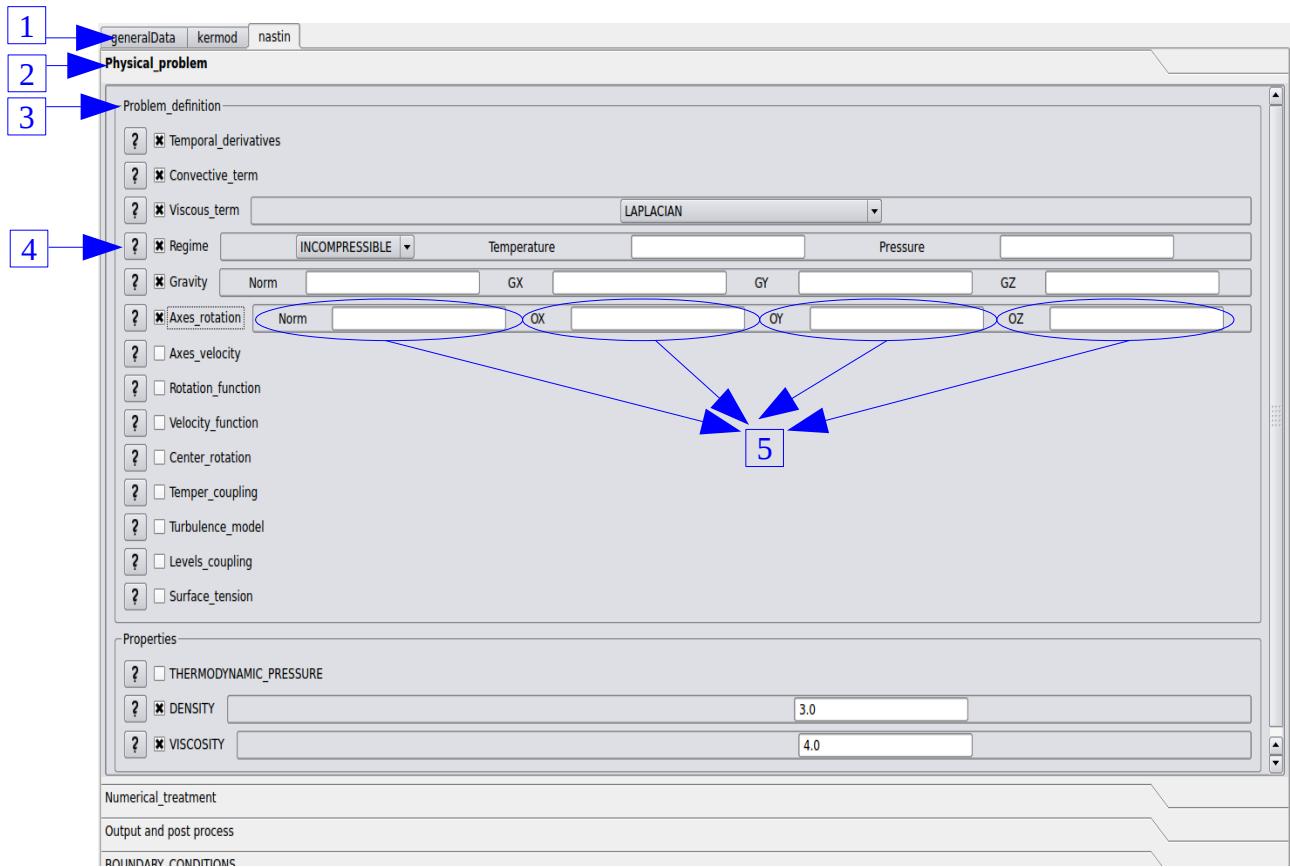


Automatización GUI ALYA

Estructura de la Gui



La Gui de Alya se compone de una estructura jerárquica, siguiendo la numeración de la figura:

- 1. Modules y services:** Por cada modulo cargado se tiene una pestaña horizontal, en la figura se tiene seleccionado el modulo de Nastin.
- 2. Group:** Cada modulo se descompone en varios grupos, para cada grupo se tiene una pestaña vertical. En la figura se tiene seleccionado el Physical_problem
- 3. SubGroup:** Este es opcional, cada grupo se puede descomponer a su vez en varios subgrupos, para cada subgroupo se dibuja un cuadro que agrupa varios inputs. En la figura se ha indicado el Problem_definition
- 4. InputLine:** Se corresponde con cada “línea” de parametros de Alya, cada una tiene el botón de ayuda y se puede activar o desactivar. En la figura se ha indicado el inputLine de “Regime”
- 5. InputElement:** Cada inputLine se puede dividir en varios inputElements, que son el nivel más bajo de jerarquía, cada uno de ellos se corresponde con un input básico de datos. En la figura se han indicado los 4 input elements de “Axes_rotation”, que son 4 inputs de texto llamados “Norm”, “Ox”, “Oy” y “Oz” respectivamente. Básicamente hay 4 tipos de inputs:
 - Combos
 - Inputs de texto sin label

- Inputs de texto con label
- Listas de inputs

Documentación a añadir a Alya para la GUI

Cada modulo en las subrutinas reaphy, reabcs, reageo y reaous tiene comentarios que generan la documentación automática, para que esos comentarios permitan también generar la Gui hay que añadir más texto.

Hay que añadir los elementos suficientes para representar la estructura de la Gui descrita en el paso anterior.

Todos los comentarios que afectan a la GUI empezarán con:

!ADOC[g]>

Con la 'g' de gui.

A continuación se indicará el tipo de elemento de la GUI, con dos caracteres, que puede ser:

- **GR:** Group
 - Su formato es GR#[NOMBRE_GRUPO]#[TIPO_GRUPO]
 - TIPO_GRUPO: OUPOS, BOUCO
- **SG:** Subgroup
 - Su formato es SR#[NOMBRE_SUBGRUPO]#[TIPO_SUBGRUPO]#[CHECKEABLE]
 - TIPO_SUBGRUPO: ALGEBRAIC_SOLVER
 - CHECKEABLE: Si se pone entonces el subgroup entero se puede habilitar/deshabilitar
- **IL:** InputLine
 - Su formato es IL#[NOMBRE_INPUTLINE]#[TIPO_ELEMENTO#TIPO_VALOR]
 - TIPO_ELEMENTO: COMBO, EDIT, LIST
 - TIPO_VALOR: REAL, INT, STRING o lista de opciones para combos: OP1|OP2|...
- **IE:** InputElement
 - Su formato es:

IE#[NOMBRE_ELEMENTO]#[TIPO_ELEMENTO#TIPO_VALOR#[VALOR_DEFECTO]]
 - TIPO_ELEMENTO: COMBO, EDIT, LIST
 - TIPO_VALOR: REAL, INT, STRING o lista de opciones para combos: OP1|OP2|...

Que se corresponde con los elementos que conforman la Gui.

Para distinguir entre InputLine/InputElement básicas y avanzadas, delante de las avanzadas se introduce:[adv]

Por ejemplo, si se tiene la siguiente documentación del grupo physical_problem:

```
! ADOC[0]> $-----
! ADOC[0]> $ Physical properties definition
! ADOC[0]> $-----
! ADOC[0]> PHYSICAL_PROBLEM
    ! ADOC[1]> PROBLEM_DEFINITION
        ! ADOC[2]> REGIME:      INCOMPRESSIBLE | LOW_MACH [TEMPERATURE= real or PRESSURE= real]
        ! ADOC[d]> REGIME:
            ! ADOC[d]> Regime of the flow. If the flow is low mach, additional information is required.
    ! ADOC[1]> END_PROBLEM_DEFINITION
    ! ADOC[d]> END_PROBLEM_DEFINITION
    ! ADOC[d]>
! ADOC[0]> END_PHYSICAL_PROBLEM
```

Los tags que hay que añadir para la Gui serían:

```
! ADOC[0]> $-----moros i cristians programa festa pollença---
! ADOC[0]> $ Physical properties definition
! ADOC[0]> $-----
! ADOC[0]> PHYSICAL_PROBLEM
! ADOC[g]> GR#PHYSICAL_PROBLEM
    ! ADOC[1]> PROBLEM_DEFINITION
    ! ADOC[g]> SG#PROBLEM_DEFINITION
        ! ADOC[2]> REGIME:      INCOMPRESSIBLE | LOW_MACH [TEMPERATURE= real or PRESSURE= real]
        ! ADOC[g]> [adv]IL#REGIME
        ! ADOC[g]> IE##COMBO#INCOMPRESSIBLE|LOW_MACH#LOW_MACH
        ! ADOC[g]> IE#TEMPERATURE#EDIT#REAL#5
        ! ADOC[g]> IE#PRESSURE#EDIT#REAL#6
        ! ADOC[d]> REGIME:
            ! ADOC[d]> Regime of the flow. If the flow is low mach, additional information is required.
    ! ADOC[1]> END_PROBLEM_DEFINITION
    ! ADOC[d]> END_PROBLEM_DEFINITION
    ! ADOC[d]>
! ADOC[0]> END_PHYSICAL_PROBLEM
```

Tipos de inputs y equivalencias soportados

Aquí se especifica todos los tipos de input que soporta la Gui, si se introdujera una nueva forma de input en Alya, que no existe aquí, la Gui no se generará correctamente.

- Permite 3 subniveles: Group, subgroup y inputLine
- Para el nivel de subgrupo se puede especificar el nombre o no, ejemplo:

- El grupo de physical_problem tiene el subgrupo “problem_definition”
- El grupo de “NUMERICAL_TREATMENT” tiene un solo subgrupo sin nombre
- Para el nivel de subgrupo se puede indicar un valor, por ejemplo, el grupo de “NUMERICAL_TREATMENT” tiene un subgrupo llamado “ALGORITHM” Que a su vez, tiene un valor llamado “SCHUR_COMPLEMENT”:

```
! ADOC[1]> ALGORITHM: SCHUR_COMPLEMENT
! ADOC[2]> SOLVER: ORTHOMIN | RICHARDSON , MOMENTUM_PRESERVING
! ADOC[2]> PRECONDITIONER: DT | TAU
! ADOC[2]> TAU_STRATEGY: CODINA | SHAKIB | EXACT
! ADOC[2]> ELEMENT_LENGTH: MINIMUM | MAXIMUM
! ADOC[2]> CORRECTION: OFF | CLOSE | OPEN
! ADOC[1]> END_ALGORITHM
```

- Para el nivel de inputLine se aceptan de dos tipos:

- inputLine simples:
 - Ejemplo: TEMPORAL_DERIVATIVES: ON | OFF
- inputLine complejos:
 - Ejemplo: FORCE_TERM: OFF | MATERIAL

- Los tipos de input elements soportados son:

- Inputs de texto con etiqueta



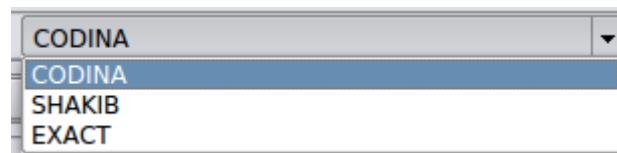
Ejemplo: GRAVITY: NORM= real, GX= real, GY= real, GZ= real

- Inputs de texto sin etiqueta



Ejemplo: CENTER_ROTATION: real1, real2, real3

- Combos



Ejemplo: VISCOSITY_TERM: OFF | LAPLACIAN | DIVERGENCE | COMPLETE

- Listas

Materials			
	1	2	3
1	1	1	1,2,3
2	2	1	4,6,1,4,7

Ejemplo:

FORCE_TERM:

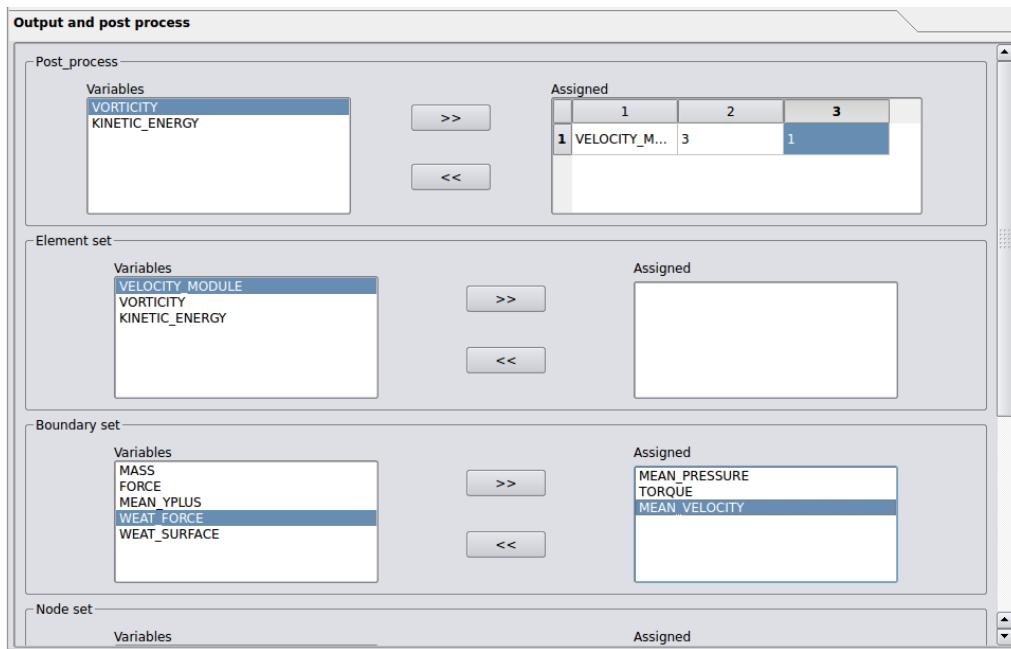
```
MATERIALS=int1, MODEL=WAKE, PARAMETERS=real1,real2,...  
...  
END_FORCE_TERM
```

Tipos de grupo

Hay grupos enteros que no siguen exactamente los tipos de input definidos en el punto anterior, además esos grupos son exactamente iguales en todos los modulos, lo que se ha hecho en este caso es crear un tipo de grupo.

- OUPOS groupType

El grupo OUTPUT_&_POST_PROCESS, que se lee con el fichero reaos, es igual en todos los módulos, además no sigue el estandar de inputLine e inputElements, sino que consiste en varias listas editables cuyos elementos provienen de otra lista. En la Gui se ha creado un tipo OUPOS que ya incluye todos los inputs automáticamente, sin tener que definirlos:



Lo único que se tiene que definir para este caso son los posibles valores de las listas de variables.

La documentación a añadir al reaos sería:

```
! ADOC[0]> $-----  
! ADOC[0]> $ Output and postprocess  
! ADOC[0]> $-----
```

```

! ADOC[0]> OUTPUT_&_POST_PROCESS
! ADOC[g]> GR#OUTPUT_&_POST_PROCESS#OPOS
! ADOC[1]> POSTPROCESS XXXXX, STEPS=int1 [, FILTER=int2]
! ADOC[g]> OPOS#postProcess#VELOCITY_MODULE|VORTICITY|KINETIC_ENERGY
! ADOC[d]> POSTPROCESS:
! ADOC[d]> Postprocess variables on nodes at each int1 time steps. the name of the file
! ADOC[d]> where the variables is stored at time step nnn is: <tt> sample-XXXXX-00000nnn.post.alyabin</tt>.
! ADOC[d]> A fliter can be applied to the variable. Filters are defined in kermod.

.
.
.

! ADOC[g]> OPOS#elementSet#VELOCITY_MODULE|VORTICITY|KINETIC_ENERGY
! ADOC[g]> OPOS#boundarySet#MEAN_PRESSURE|MASS|FORCE|TORQUE|MEAN_YPLUS|
MEAN_VELOCITY|WEAT_FORCE|WEAT_SURFACE
! ADOC[g]> OPOS#nodeSet#VELOX|VELOY|VELOZ|PRESS|YPLUS
! ADOC[g]> OPOS#witnessPoints#VELOX|VELOY|VELOZ|PRESS

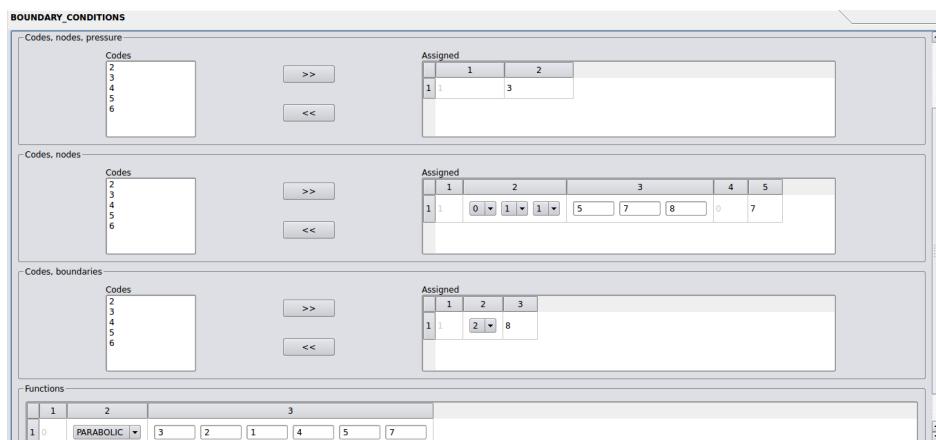
```

- BOUCO groupType

El grupo BOUNDARY_CONDITIONS, que se lee desde el fichero reabcs, es igual en todos los módulos, además no sigue el estandar de inputLine e inputElements, sinó que consiste en varias listas que se generan a partir de otras listas.

Además, tiene la peculiaridad que las listas base son los códigos de boundary conditions que están en la definición de la malla, por lo que estos deben ser cargados dinámicamente desde el fichero geo.dat del problema. Por lo que es imposible crear automáticamente todo esto a partir de unos comentarios en la documentación.

Se ha creado una pantalla de la GUI que hace todo esto automáticamente:



Para crear este grupo de la GUI solo hace falta especificar su tipo en la definición:

```
! ADOC[0]> $-----  
! ADOC[0]> $ boundary conditions  
! ADOC[0]> $-----  
! ADOC[0]> BOUNDARY_CONDITIONS  
! ADOC[g]> GR#BOUNDARY_CONDITIONS#BOUCO  
! ADOC[d]> BOUNDARY_CONDITIONS:  
! ADOC[d]> In this section, the boundary conditions are defined.
```

Tipos de subgrupo

Se ha observado que algunas estructuras de subgrupo se repiten dentro de un mismo grupo, como por ejemplo a la hora de definir los diferentes solvers(ALGEBRAIC_SOLVER) en el “NUMERICAL_TREATMENT”:

```
! ADOC[1]> XXX_EQUATION  
! ADOC[2]> ALGEBRAIC_SOLVER  
! ADOC[2]> INCLUDE ./sample-solver.dat  
! ADOC[2]> END_ALGEBRAIC_SOLVER  
! ADOC[1]> END_XXX_EQUATION  
! ADOC[d]> XXX_EQUATION:  
! ADOC[d]> Define the algebraic solver options for equation XXX.  
! ADOC[d]> XXX: MOMENTUM | CONTINUITY | HYDROSTATIC_STATE
```

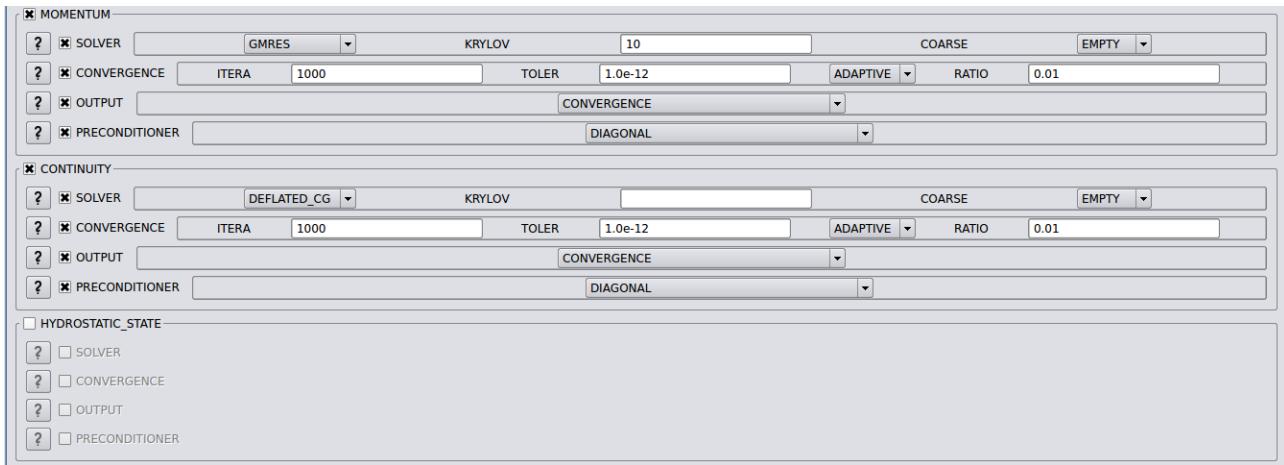
Aquí el mismo tipo aparece 3 veces: para MOMENTUM, CONTINUITY y HYDROSTATIC_STATE.

Además estos subgrupos tampoco siguen el estándar de los inputLines e inputElements.

Así que se han decidido crear en la GUI los tipos de subgrupo, donde en lugar de definir cada uno de los inputs en la GUI simplemente se llama a un subtipo que los pinta todos automáticamente.

- ALGEBRAIC_SOLVER subGroupType

En este caso, la gui que se crea cuando se define este subgrupo sería:



- La forma de introducir esto en la documentación sería:

```
! ADOC[1]> XXX_EQUATION
! ADOC[2]> ALGEBRAIC_SOLVER
! ADOC[2]> INCLUDE ./sample-solver.dat
! ADOC[2]> END_ALGEBRAIC_SOLVER
! ADOC[1]> END_XXX_EQUATION
! ADOC[g]> SG#MOMENTUM#ALGEBRAIC_SOLVER#CHECKEABLE
! ADOC[g]> SG#CONTINUITY#ALGEBRAIC_SOLVER#CHECKEABLE
! ADOC[g]> SG#HYDROSTATIC_STATE#ALGEBRAIC_SOLVER#CHECKEABLE
```